DOCUMENT RESUME

ED 428 695 IR 019 356

AUTHOR Maly, K.; Overstreet, C. M.; Gonzalez, A.; Denbar, M. L.; Cutaran, R.; Karunaratne, N.

TITLE Automated Content Synthesis for Interactive Remote Instruction.

PUB DATE 1998-06-00

NOTE 7p.; In: ED-MEDIA/ED-TELECOM 98 World Conference on Educational Multimedia and Hypermedia & World Conference on Educational Telecommunications. Proceedings (10th, Freiburg, Germany, June 20-25, 1998); see IR 019 307. Figures may not reproduce clearly.

PUB TYPE Reports - Descriptive (141) -- Speeches/Meeting Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.

DESCRIPTORS *Computer Assisted Instruction; *Computer Mediated Communication; Computer Oriented Programs; *Computer System Design; *Courseware; *Distance Education; Educational Technology; Higher Education; Interaction; Navigation (Information Systems); Teleconferencing; World Wide Web

IDENTIFIERS *Learning Environments; Old Dominion University VA; Paradigm Shifts; Technology Implementation; *Virtual Classrooms; Web Pages

ABSTRACT

This paper describes IRI (Interactive Remote Instruction), a computer-based system built at Old Dominion University (Virginia) in order to support distance education. The system is based on the concept of a virtual classroom where students at different locations have the same synchronous class experience, using networked computers to communicate through video, audio, and tool sharing. During a synchronous session, all individual streams of action are recorded with timing points. This information is synthesized and presented as a set of World Wide Web pages that can be used at a later time to review any portion of the lecture using the Web navigation pages as an index to all class activities. How the learning paradigm is shifting as a function of technological advances is considered. Relevant collaboration, videoconferencing, Web, and cross-platform tools are described, and the ideal environment (i.e., a set of integrated tools, which provide support for the learning experience) is discussed. The recording architecture is summarized, including recording files, playback architecture, and recording playback synchronization. Technical issues related to implementation are addressed, including multiple-user steering, and content synthesis and presentation. Two figures present the basic IRI software and recording server architecture, and the replay control panel. (DLS)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED 428 695

Automated Content Synthesis for Interactive Remote Instruction

K. Maly, C. M. Overstreet , A. González, M. L. Denbar, R. Cutaran, N. Karunaratne
Computer Science Department, Old Dominion University, Norfolk, VA 23529-0162 USA
Tel: 757-683-4545, Fax: 757-4900, E-mail: {maly,cmo}@cs.odu.edu

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☐ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ABSTRACT: At Old Dominion University, we have built IRI, Interactive Remote Instruction, a computer-based system to support distance education. We have used IRI to teach several classes at sites up to 200 miles apart. In this paper we describe the extensions we are making to use IRI in both synchronous and asynchronous modes. We describe the architecture used to support recording of all class activities for future replay, and describe the interface which can be used to steer an IRI session.

1. Introduction

In higher education, distance learning is becoming key as more nontraditional students take courses. Two successful distance learning methods are TV-based and Web-based courses [Fox et al. 1995]. In the first method a course is transmitted to remote sites. In Webcourses, material is placed in a multimedia, hyper-linked format which students access through the Internet at any time. Many such courses allow students to interact with the system for progress assessment or to execute tools for acquiring skills. In their pure form the two approaches are on opposite ends of a synchrony scale. TV courses have added asynchronous capabilities by providing videotapes of a lecture that can be viewed at any time. However, tapes do not provide an index that enables students to navigate the material easily. Webcourses can provide excellent maps of their contents and are ideally suited for review at one's own pace and abilities. The disadvantage of Webcourses is the high cost to create a good one. It may take $100,000 and six months for a team of web experts, content providers, and graphics designers to put together a new course.

We propose a new approach intended to combine the best of the two methods. It is based on the concept of a virtual classroom where students at different locations have the same synchronous class experience. Each student uses a computer as his/her window into the classroom. Students use networked computers to communicate through video and audio and tool sharing. IRI (Interactive Remote Instruction), a system we have developed over the past few years, supports this approach. In this paper we describe how IRI can be used for both synchronous and asynchronous participation. Current IRI capabilities are described more fully in [Maly et al. 1997]. IRI records all audio and video, who speaks, whose video is shown and what tools are running when. The concept is simple: during a synchronous session, record all individual streams of actions with timing points. This information is synthesized and presented as a set of Web pages that can be used at a later time to review any portion of the lecture using the Web navigation pages as an index to all class activities.

Section 2 of this paper provides some background on technologies including a summary of IRI. In section 3 we describe the architecture of the recording and the replaying features of IRI followed by the design and implementation of the synthesized Web pages in section 4.

2. Tools and Environments

In this section we describe how the learning paradigm is shifting as a function of technological advances. Since ours is a technological research perspective we present a characterization based on technology and educational theories. First we summarize the relevant technologies, followed by a characterization of learning paradigms and then describe how various application level tools and environments support learning. The ideal paradigm we propose, and believe current technology can support, is one which scales well, is symmetric, allows for both asynchronous and synchronous use, has high perception quality, is highly interactive, does not require students to be collocated, allows learning tools to be shared, is cost effective, and can be used in groups or individually as appropriate for a set of learning objectives.

PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
G.H. Marks
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

IR019356
BEST COPY AVAILABLE

The tools we describe in this section are the technology tools and environments, which enable new learning paradigms. We categorize tools by which feature in a learning experience they support; for example, a video conferencing tool supports group interaction. An environment is a set of integrated tools, which provide support for most of the learning experience of a student.

Collaboration tools allow learning tools to be shared among class members in the sense that anyone can operate them. In the Unix arena an example is XTV [Wahab 1991], which allows the sharing of X programs. In the NT world, Netmeeting lets a group share win32 programs. These tools use TCP/IP and thus do not scale well and cannot handle large groups. No current tool allows the sharing of arbitrary learning tools across operating system platforms, but tools are being developed which will allow the sharing of specific tools across platforms. For instance, WEB-4M is a JAVA based tool, which allows the sharing of a white board and slide tool.

Video conferencing tools exists both over IP networks and ISDN based telephone networks. Vic and vat, a suite developed by van Jacobson, was developed for large audience as an asymmetric tool, i.e., for broadcasting to thousands with few sources. The tool is based on the Mbone and includes a whiteboard. SmartStation proprietary tool works over telephone networks with switched ISDN lines. It does not scale well and can become expensive because typically at least three ISDN lines (192K) are necessary for each participant to support an acceptable level of perception. None of the video conferencing tools support tool sharing.

Webtools are used to support Webcourses. At a basic level are the editors included with Netscape and Internet Explorer for creating multimedia web pages and groupware such as e-mail, chatrooms, newsgroups, and document managers. At an advanced level, Domino uses a secure webserver and provides a framework for writing applications that support group activities. Authoring tools help transform the educational content of a course into an interactive webcourse to the point that no specialists are needed. Other tools support the assessment part of the learning, for instance, QW (Oregon State University) allows teacher to create quizzes, administers and grades the multiple-choice components. Web-based whiteboards and specialized presentation tools are common. C-browsing is a coordinated browser, which allows a group of participants to visit the same page on the Web together. Anyone in the group can click on a link and all participants can see the result.

Cross-platform tools are being developed for the two dominating platforms NT and Unix. The identification of Unix with academe and NT with home PCs is steadily eroding, hence the need to run a tool written for one platform on another platform. The best known tool for running X programs on a PC is Exceed which basically creates an X server under NT and allows any X program running on a Unix box to be displayed and manipulated on the PC running NT. Citrix provides the opposite: it runs on an NT server and any program running under NT can be displayed and manipulated by a Citrix client running on a Unix box. All of these tools, however work on a one-to-one basis. That is, it is impossible for a group of users on Unix boxes to see a tool running on a PC.

A successful environment is one where the end user has to perform few operations to operate the environment. TV based distance learning is a good example in this category. The student has only to go to a site and learn how to operate the audio button to ask a question. The entire system is run by technicians who ensure that the sending, receiving equipment and the cameras operate smoothly. The teacher does have to be trained in what can and cannot be presented in this medium and how to make effective use of cameras and audio channels. Similarly, environments have been created for Webcourses that integrate administration tools, content generation, and monitoring of students' progress; WEB-4M and the environment at Virginia Tech are good examples. CU-Seeme is an environment that combines video conferencing, whiteboard, e-mail, document sharing and other tools into one environment.

IRI is an environment which is being developed with the goals of the ideal paradigm described above. Currently, this environment solves the scaling problem for up to an order of 100 users while maintaining tool-sharing capabilities by the use of reliable multicasting, specifically, RMP. IP-multicast by itself does not support reliable transport that is essential when transferring data. IRI in its current state does not support true interoperability, i.e., a user can be at either an NT computer or a Unix box. X tools can be shared to any platform but not PC tools. Because of its use of multicasting it is totally symmetric. IRI supports both synchronous and asynchronous learning by recording live synchronous sessions in their totality (audio, video, and tool traffic) and automatically synthesizes the content to the asynchronous viewer.

3. Recording Architecture

In this section we briefly describe the components added to the IRI architecture to support recording and playback of IRI sessions. Two servers were added to the architecture: a Recording Server (RS) and a Playback Server. In general both resembles the Remote Instruction Server (RIS) described in [Wahab et al. 1996]. The architecture for recording in IRI is made up of a web-based interface, a session recording control (SessRec), a

recording process for each type of stream, such as audio, video, slide show tool, and a Reliable Multicast Protocol Server (RMPS), as shown in [Fig. 1].

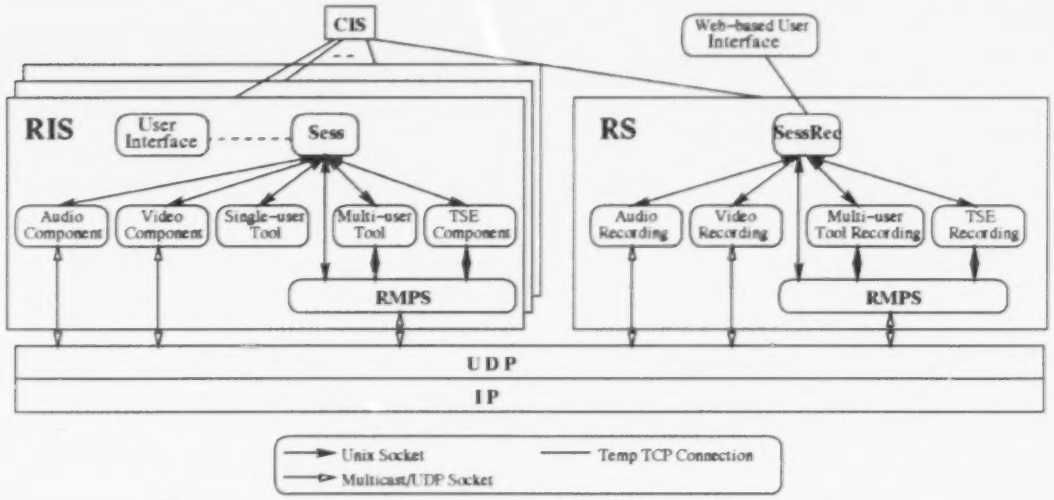

Figure 1: Basic IRI Software Architecture and Recording Server Architecture

A web-based interface allows the session leader to turn recording on or off at any time (for example during a class break). It establishes a TCP connection with SessRec and sends messages to control recording. The session recording control process is started like any other session control process of a RIS. Then it spawns the other recording components. As far as the rest of the IRI session is concerned, the recording server is a passive participant. SessRec does not bring up any interface, but waits for the connection of the web-based interface. In addition to the normal operation of an IRI session, SessRec stores the allocation of the media resources in a file, as the session progresses. The audio recording process joins the audio channels following the same protocol already implemented between session control and audio component of RIS. The major difference with the audio component of RIS is the redirection of the output to files rather than the audio device. In addition to audio streams that are stored in μ-law format, the audio recording process records timing information carried in audio packets to take into account silences and packet losses due to the unreliable multicasting.

The video recording process works like a video receiver process in RIS. However, instead of decompressing and displaying video streams on IRI windows, it stores them to a local disk using a video size of 320x240 pixels. There is no need for decompressing most of the video streams, but the presenter video (640x480 pixels) has to be decompressed, reduced, and then compressed again for storing. Like the audio recording process, this process also stores both the video stream and the necessary timing information to overcome gaps due to video frames losses. IRI has a number of specialized tools that were written especially for IRI. A modified version of each works in recording mode to store the important events associated to that tool. Later, during review, another version of this tool running in playback mode reads the events from a file and replays them accordingly to each tool operation. The Tool Sharing Engine (TSE) of IRI, based on the XTV system, allows sharing of X windows' applications collaboratively. The recording server runs a modified version of it, the tool-sharing engine-recording tool, to store the events related to any application running under this component.

3.1 Recording Files

Two layers of files are stored for each recorded media: a resource allocation layer and a data stream layer. The former is recorded by the session recording control process. It stores the users' names and times at which the resource was used. The latter layer of files stores the timing and data associated to each resource. One file keeps the timing information of the stream and pointers to the actual data that are saved in another file using standard formats, such as Sun μ-Law for audio files and GIF for slides. For example, in the case of audio, session recording control stores the time and names of those who have been granted an audio channel and those who release it as well. In addition, the audio recording process records the initial time of every recorded audio block

and the offset where that block is stored within the data stream file. Thus, at review time a corresponding audio playback process can access the audio data and its timing to replay it as it was heard by the participants of the synchronous session. For video, we do not record individual user's video streams (whose total is of the order of participants) but the windows' videos seen by any participant (a total of 4 in the current IRI implementation). For each window's video, a session recording process stores when and who took over that video resource (video allocation file for that window), and a video recording process stores the video data stream and its timing information. Slide show tool events and slides are recorded with a similar scheme to audio and video.

3.2 Playback Architecture

The architecture for playback in IRI is similar to the recording architecture. For every process in the recording architecture, there is a complementary process in the playback. While recording processes receive and write media streams to disk, playback processes read these streams from disk and send them to an IRI session. A web-based interface allows participants to see recorded IRI sessions and the resources that have been recorded for each person. Then users can select the streams to be played back. This interface connects to a session playback control process (SessRep) and sends messages to initiate, resume, or stop playback. The communication between the interface and SessRep is via TCP sockets.

Session playback control serves requests sent by the review interface, allocates and releases resources for streams being played back, and communicates to each playback process to start and stop their operation. A session control panel (see Section 4) is used to start the session playback control process (SessRep) just like any other session control process. Then SessRep spawns the other playback processes following the same protocol that session control uses. As far as the whole IRI session is concern, SessRep is like another session control with an interface controlled by a review user. Like session recording control, SessRep does not bring up an interface, and it waits for messages from the web-based interface. SessRep displays labels and graphical indications for playback streams as opposed to live session streams.

The audio playback process operates like other IRI audio processes but reads recorded audio data. This greatly effects timing. While intra-stream synchronization during recording is controlled by a fixed sample rate of the audio input device, some mechanism needs to be provided for playback. The audio playback process inserts pauses to compensate the removal of silent periods and the loss of audio packets during recording. The video playback process works like a video sender of a normal IRI session, but instead of capturing video frames from a camera it reads them from a file and send them over the network. Like audio and video, there are other processes in charge of playback for each IRI multi-user tool, such as slide show tool, and a version of the tool-sharing engine tool for playing back the applications that run under this component, such as Netscape and emacs.

3.3 Recording Playback Synchronization

Timestamps are used for intra- and inter-stream synchronization. A data unit timestamp is the time in milliseconds since the IRI session begins. Video and audio streams are the most critical in terms of synchronization requirements. They carry timestamps, attached at the sender host to each application data unit (packets for audio frames for video). While video streams have a clear timestamp used for recording, it is not clear how to assign a timestamp to the audio stream coming out of the audio-mixing algorithm. Our approach uses the timestamp associated to the loudest audio stream being mixed. By doing this we keep synchronized the video/audio pair that has the attention of the participants.

In order to ensure that each RIS computes local timestamps from a common starting time for the class, we use the first leader's event sent reliably to synchronize the starting time of the class in every participant session. Another reason for a global synchronization is the need for a unique time line for resources allocation. Unlike video and audio streams, all other IRI resources set timestamps at the receiver, so we record what everyone sees as delivered by RMPS. Thereby, we need the recording server to be in sync with any possible audio or video stream generator (i.e., anybody).

Every playback process takes the local machine time and timing information associated to each data unit to synchronize the replying of its stream; thus intra-stream synchronization is accomplished. Inter-stream synchronization is achieved by letting session playback control send a common playback starting time to all the processes involved. This way specific synchronization techniques are not required within dynamically changing playback processes set.

4. Implementation

In this section, we focus on the technical issues involved in building a Web-based interface to steer an IRI session in which we replay portions of previously recorded sessions. In the past we have used a Motif interface to steer an IRI session and to add resources needed for a session. Since we are in the process of developing a cross-platform implementation of IRI we decided to switch to a Web-based controller as a first step. First we discuss how we enable multiple users to steer and monitor a session securely and how we integrate their browsers with the IRI interface. In the second part, we present the content synthesis and presentation to the user for selecting arbitrary points of starting a replay.

4.1 Multiple-User Steering

IRI is based on reliable multicasting because of the number of students participating in a session. However, we expect only a few people, typically the teacher and an assistant, to be involved in the steering and monitoring process. Therefore, we can use normal, TCP/IP based, Web server-to-browser communication to coordinate the browsers of the controller group. The entire interface is a set of CGI scripts/programs, JAVA scripts and applets that access a protected directory on the server side and communicate with IRI through Unix sockets and with direct access to the IRI file tree. Each Web page presented to the user requires a proper authentication token that is only obtained by password authentication.

To start a session a teacher goes to the appropriate web page and authenticates herself. After specifying on subsequent pages the configuration for that session (machines to be used, servers, lesson plan) and the location and identities of specified controllers and monitors (defaults are available for all of these), she starts the session by sending a set of messages to the IRI process. We use the interface button for "class management" of IRI to connect to these browsers and make them active for all the controllers and monitors. Thus, at any time of the session, the teacher, for example, can click on "class management" and bring the browser to the foreground and click on "start replay" and proceed from thereon as described below. Similarly, the monitor can bring up his Web page at any time and proceed to monitor the session using the pages provided.

4.2 Content Synthesis and Presentation

Assume that a group of students have started a session to review a previously recorded session. The group leader started the session on his PC in an office and has decided to steer the session himself on a machine in the IRI classroom that has no monitor. After logging on the machine, he uses regular IRI features to communicate with the other students. At some point he brings his control browser in the foreground and clicks "Activate replaying." A cgi script will ask him to identify what session to replay and when that is specified, present him with a synopsis of that session through JAVA applets as shown in [Fig. 2]. The JAVA applet knows already what session is to be replayed and presents a list of participants who spoke or had their image present and at what times of the session. In addition the leader can select the level of abstraction the presentations should be given. The choices are class, presentation, and slides. The first one implies that the applet will only show from which classes the leader of the recorded session used presentations. In the second choice, the applet will present the times each different presentation was on the screen during the session. In the example shown in [Fig. 2], the leader selected the third option and was given all the slides visited during the entire recorded session. The key is that each of these applets' windows can be left on the screen and at any time the leader can click on any of the time intervals to start replay at a particular point of time. These windows provide the synopsis of the session in different forms and depending on how a person remembers best, she can choose whatever stream is most appropriate.

Once the leader clicks on a particular time interval, at a spot estimated, the applet computes from the two shown endpoints of the interval the time the user intended and sends a message to the replay IRI process. This one in turn, computes which IRI process was active at that time and sends messages to activate the appropriate processes (audio, video, or presentation tool). The replaying process selects a window on the IRI interface, tags it as being a recorded window, and plays as long as the next event happens, e.g., stream ends or next slide is shown.

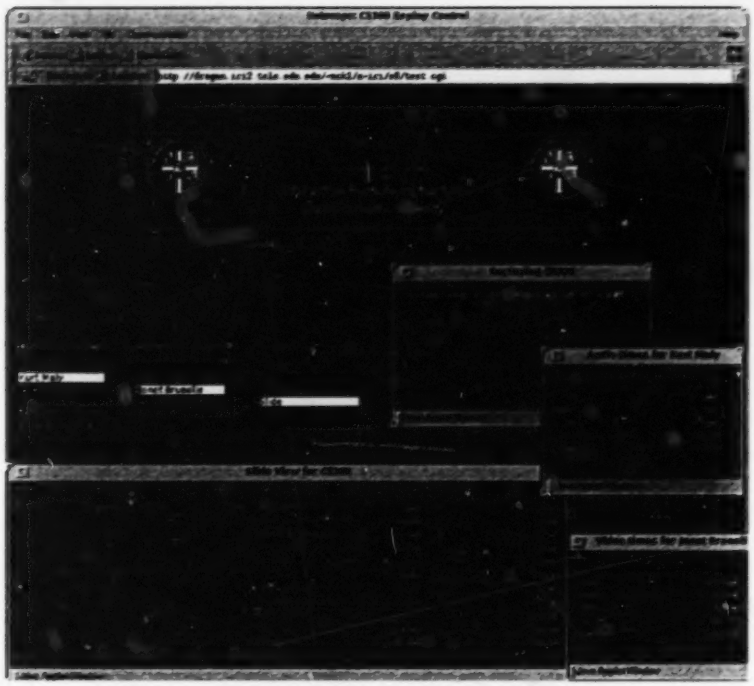

Figure 2: Replay Control Panel

5. Status and Future Work

At this point we have the first version of recording and replaying implemented and tested; it is not yet being used in an actual class environment because not all the features shown in [Fig. 2] are complete. At the next level we will use the same process for the remaining special IRI tools such as survey, exam, and coordinated browsing. The next major step will be to synthesize and abstract the recording of arbitrary X tools that were used in a session. Since we have no control over these tools, we will not be able to provide any summary information except screen images at selected time points. But even with these images as index to a time interval we have the problem of starting an X tool in the middle of its execution. We are investigating the concept of fast forward for the playing of X tools which will elide X commands which are not relevant after a certain time.

6. References

[Wahab 1991] Abdel-Wahab, H., and M. Feit (1991). Communications for Distributed Applications & Systems. *XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration*. IEEE TriComm '91, Chapel Hill, North Carolina, 159-167.

[Wahab et al. 1996] Abdel-Wahab, H., K. Maly, A. Youssef, E. Stoica, C. M. Overstreet, C. Wild, and A. Gupta (1996). Proceedings of the IEEE Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. *The Software Architecture and Interprocess Communication of IRI: an Internet-based Interactive Distance Learning System*, Stanford University, 4-9

[Fox et al. 1995]. Fox, E. and L. Kieffer (1995). Multimedia curricula, courses and knowledge modules. *ACM Computing Surveys*, 27(4):549-551, Dec. 1995.

[Maly et al. 1997]. Maly, K., H. Abdel-Wahab, C. M. Overstreet, C. Wild, A. Gupta, A. Youssef, E. Stoica, & E. Al-Shaer (1997). Interactive Distance Learning Over Intranets. *IEEE Internet Computing*, 1 1, 60-71.

BEST COPY AVAILABLE